

FACHHOCHSCHULE HEILBRONN

Fachbereich Software Engineering

Einführung in Digital Signal Processing

FACHBEREICH SOFTWARE ENGINEERING

Einführung in Digital Signal Processing

Fachhochschule Heilbronn
Max-Planck-Straße 39 • 74081 Heilbronn
Telefon 07131/50 4 - 0 • Fax 07131/25 24 70

Inhaltsverzeichnis

Kapitel
1

Motivation und Einleitung

„Auch eine Reise von tausend Meilen beginnt mit einem Schritt."

Chinesisches Sprichwort

Dieses Dokument gibt einen Überblick über die elementaren Funktionen der digitalen Signalverarbeitung im Allgemeinen und der *Texas Instruments TMS320C50* DSP – Karte. Es soll Sie bei Ihren ersten Schritten mit der Texas Instruments Karte unterstützen und gegebenenfalls eine Hilfestellung bei eventuell auftretenden Fehlern darstellen.

Der Grund für die Erstellung dieses Dokuments waren einerseits die Aufgabe eine Projektarbeit mit der DSP – Karte durchzuführen und andererseits die Probleme, die während der Bewältigung dieser Aufgabe auftraten. Um bei nachfolgenden Projekten in der Anfangsphase die Komplexität zu verringern, setzten wir uns zum Ziel, eine gut verständliche Einführung in die digitale Signalverarbeitung zu erstellen. Anhand eines Beispielprojekts wird dem Leser eine mögliche Herangehensweise an ein ähnliches Projekt aufgezeigt.

SYMBOL-LEGENDE
Informationen
Beispiel
Auf der CD
Vorsicht

Um Ihnen die Orientierung in diesem Handbuch zu erleichtern, haben wir den Text in bestimmte Funktionsabschnitte gegliedert und diese durch entsprechende Symbole oder Icons gekennzeichnet. So können bereits beim Überfliegen des Dokuments wichtige Stellen erkannt werden. Die nebenstehende Tabelle gibt Ihnen ein Überblick über die vorkommende Symbole und erklärt deren Bedeutung.

Was ist DSP ?

Wie der Name Digital Signal Processing bereits sagt, besteht die Aufgabe von DSPs in der Digitalisierung und Verarbeitung von Signalen. Ähnlich den menschlichen Sinnen, nimmt der DSP analoge Signale auf. Da jedoch ein Prozessor nicht wie das menschliche Gehirn mit analogen Signalen umgehen kann, ist es nötig, diese Signale zu digitalisieren. Herkömmliche Computer sind ohne DSP – Unterstützung nicht dafür geeignet, da sie viele der Operationen die in den Algorithmen zur Digitalisierung enthalten sind, nicht performant ausführen können. Obwohl es

möglich ist analoge Computer zu bauen, liegt unser Interesse mehr in der Benutzung von digitalen Computern wie dem PC. Diese sind zwar besser im Umgang mit numerischen Operationen, wie z.B. in Datenbanken oder Tabellenkalkulationsanwendungen, jedoch nicht besonders für die Verarbeitung der kontinuierlichen variablen Signale unserer Umwelt geeignet.

Ein typisches DSP System

Die drei elementaren Funktionen eines DSP Systems sind:

- Wandlung der Eingangssignale
- Verarbeitung
- Ausgabe der Signale.

Die Aufgabe der Wandlung der Eingangssignale übernimmt der *Analog-to-Digital Converter* (A / D-Wandler). Dieser konvertiert die elektrischen Impulse in digitale Signale. Die Weiterverarbeitung des Signals wird von einem digitalen Computer ausgeführt und wird deshalb *Digital Signal Processing* oder kurz *DSP* genannt. Moderne DSP Systeme verwenden üblicherweise single-chip Prozessoren, die speziell für die digitale Signalverarbeitung ausgelegt sind. Diese speziellen Prozessoren werden als *Digital Signal Processors* oder kurz *DSPs* bezeichnet. Nachdem die Signale vom DSP-Chip verarbeitet wurden, liegen sie immer noch in digitaler Form vor und müssen in ein analoges Signal zurückgewandelt werden, bevor sie z.B. über einen Lautsprecher ausgegeben werden können. Dieser Verarbeitungsschritt wird vom *Digital-to-Analog Converter* (D / A-Wandler) ausgeführt.

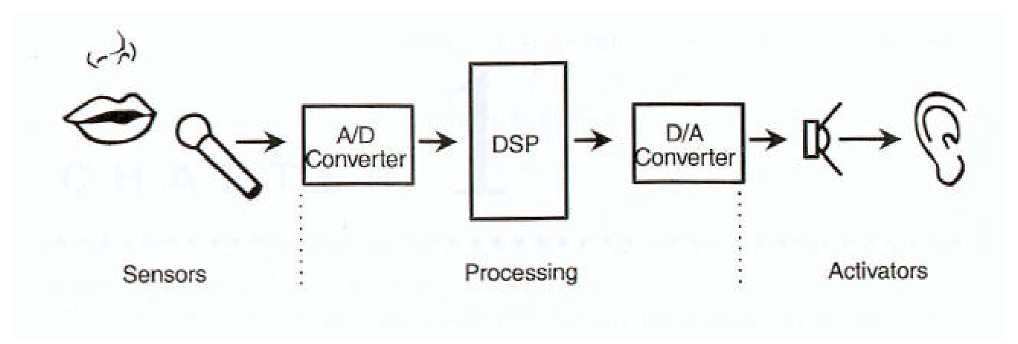


Abb. 1.1 Die komplette Wandlungs- und Verarbeitungskette

Ein typisches DSP System besteht üblicherweise aus mehreren Hardwarekomponenten, die jedoch aufgabenabhängig variieren können. Neben den oben beschriebenen A / D- und D / A-Wandlern sind die wichtigsten Elemente:

- **Central Arithmetic**

  Der Hauptprozessor auf einer DSP Karte, der für die Ausführung der meisten arithmetischen Funktionen zuständig ist.

- **Auxiliary Arithmetic Unit**

  Eine Art Koprozessor, der für die Ausführung anderer mathematischer und logischer Funktionen zuständig ist, wenn der Hauptprozessor beschäftigt ist..

- **Serielle Schnittstelle**

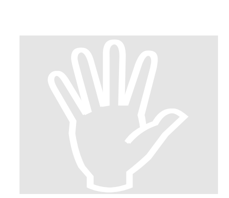
  Die serielle Schnittlstelle dient zur Kommunikation mit anderen DSP's oder PC's .

- **Speicher**

  Beim Speicher bzw. der Speicherverwaltung gibt es einen signifikanten Unterschied zur „von Neumann" – Architektur herkömmlicher Rechner. Während beim „von Neumann" – Rechner Programme und Daten in einem gemeinsamen Speicher gehalten und adressiert werden, sind bei DSPs zwei getrennte Speicherbereiche für Programme und Daten vorgesehen, die nicht nur eigenständig adressiert werden können, sondem auch getrennte Datenbusse besitzen. Der Vorteil dieser sog. „Harvard" - Architektur ist die Möglichkeit, auf Programme und Daten simultan zuzugreifen zu können.

Entscheidender Vorteil von DSP Systemen gegenüber herkömmlichen Softwarelösungen ist, dass der Prozess des Wandelns, der Verarbeitung und Rückwandlung in Echtzeit durchgeführt wird. Bei der digitalen Signalverarbeitung kommen Berechnungen der Form A = B * C + D sehr häufig vor. Normale Computer können zwar sehr schnell addieren, brauchen jedoch für die Multiplikation mehrere Rechenzyklen. DSP Chips dagegen sind so aufgebaut, dass sie die oben aufgeführte Rechenanweisungen (MAC = Multiply and Accumulate) in einem einzigen Rechenschritt ausführen können. Im Gegenzug wären sie für die Aufgaben eines Standardprozessors ungeeignet, da der Befehlssatz von DSPs größtenteils auf die Verarbeitung von digitalen Signalen beschränkt ist.

Einsatz von DSP Systemen



Heute werden in einer Vielzahl von Einsatzgebieten DSPs verwendet . So finden wir DSPs in alltäglichen Gebrauchsgegenständen. Zum Beispiel werden DSPs in der Automobilindustrie vielfältig eingesetzt. So wären heutige Annehmlichkeiten in Autos, wie Motormanagement, Klimaanlage, ABS, ESP oder Navigationssysteme ohne DSP Steuerung kaum denkbar. Aber auch in häuslicher Umgebung halten in immer mehr Einsatzgebieten DSP Systeme Einzug. Hier seien als Beispiele nur einige Einsatzgebiete genannt:

- HiFi-Anlagen, insbesondere digitales Equipment (CD, DVD, Surroundsysteme, etc.)
- Bildtelefone
- Handys
- Modems

Darüber hinaus werden auch DSPs in der professionellen Bildbearbeitung oder 3D Grafikanwendung eingesetzt. In diesen Einsatzgebieten wären typische Anwendungen Simulationen, Fingerabdruckerkennung, Qualitätskontrolle, etc.

Was ist Sampling ?



Im vorigen Abschnitt wurde bereits die Umwandlung von analogen in digitale Signale erwähnt. An dieser Stelle wird nun etwas genauer auf diese Technologie eingegangen.

Zum Arbeiten mit einer digitalen Tonaufzeichnung bedarf es die bereits erwähnten Wandler: Den A/D Konverter um sie „aufzuzeichnen" und das Gegenstück, den D/A Konverter, um sie wieder als Audiosignal ausgeben zu können.

Die Spannungsimpulse des Audiosignals, die also am Eingang der Karte anliegen, werden im Wandler abgetastet und in Zahlenwerte codiert. Die dafür gültige Bezeichnung „Pulse Code Modulation" (PCM) beschreibt das Codieren dieser Impulse.

Eine solche Abtastung rastert das komplexe Wellengebilde in Impulse und registriert für sie Amplitudenwerte, die sich als binäre Zahlencode darstellen lassen. Die Exaktheit dieser Rasterung (Auflösung) hängt von der Frequenz ab, in der das Klangereignis „nachgezeichnet" (Samplingfrequenz) wird.

Um das Messergebnis mit dem geringsten Aufwand akkurat abzutasten, sollte die Samplingfrequenz erst einmal mindestens so hoch aufgelöst sein wie die Anzahl der Amplitudenimpulse, die das Signal selbst in der Sekunde abgibt (seine eigene Frequenz). Vergleichen Sie hierzu die Ausschläge der Pegelanzeige eines analogen Kassettenrekorders.

Wir wissen allerdings vom Fourier Modell, dass die Frequenz oder Stimmung des Instrumentaltons gerade einmal über den Grundton, die tiefstgelegene harmonische definiert wird. Erst die darüber liegenden Teilfrequenzen geben dem Klang seine unverkennbaren Eigenschaften. In entsprechender Auflösung (Frequenz des Geräuschs = Samplingfrequenz) abzutasten wäre also doch nicht akkurat – eine solche Auflösung genügt noch nicht.

Diese Frage greift das Nyquist Theorem auf, welches eine optimale Abtastung eines Geräusches mit einer Mindestauflösung in der doppelten Frequenz definiert, wobei die höchsten, im Geräusch vorkommenden Teilfrequenzen nicht über der halben Samplingfrequenz liegen sollen. Für diesen Fall muss die Abtastung noch höher aufgelöst werden.

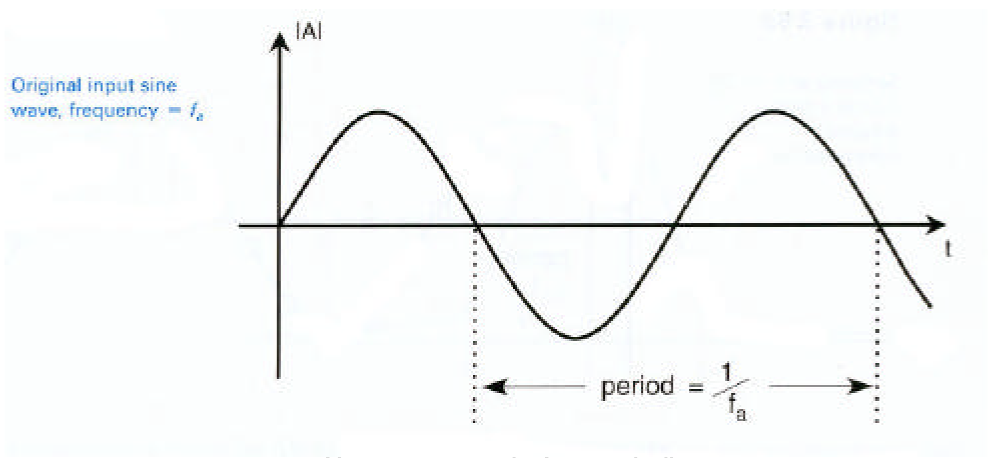


Abb. 1.2 Die zu samplende Originalwelle

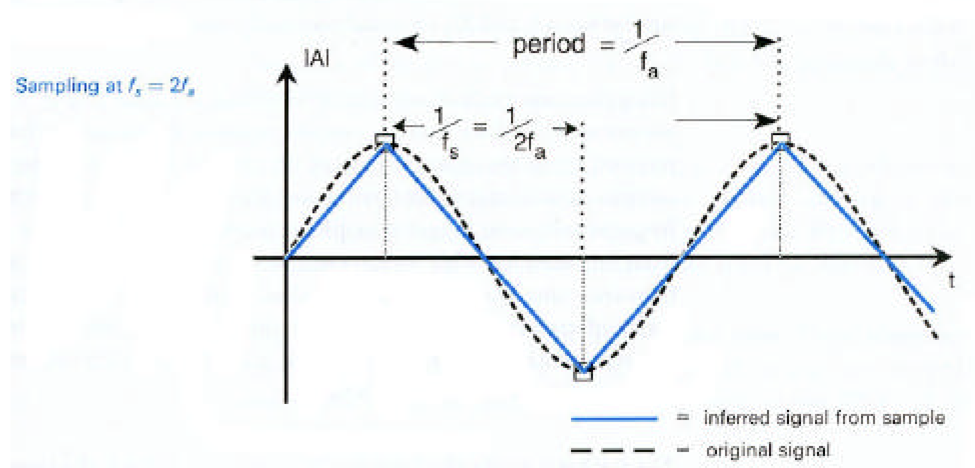


Abb. 1.3 Eine optimale Abtastung nach Nyquist

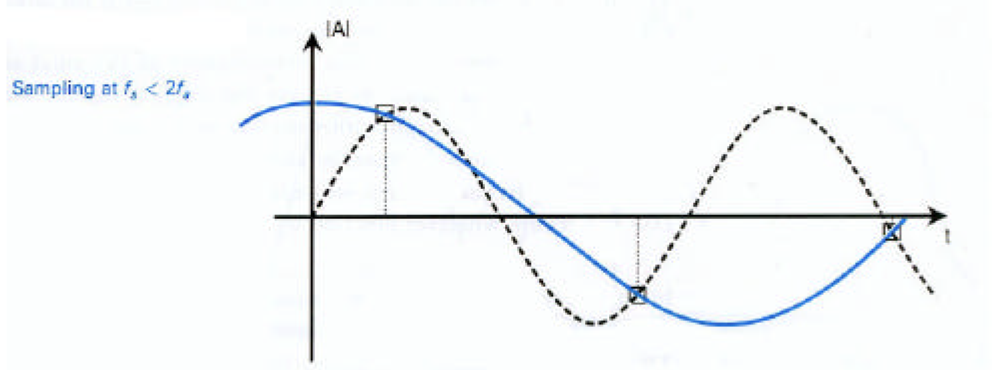


Abb. 1.4 Nicht optimale Abtastung des Signals

Es wäre jedoch ausgesprochen schwierig, wenn man jedem Samplingvorgang eine Messung der höchsten Partialfrequenz voranstellen würde. Ein solches Theorem gibt daher nur das theoretische Minimum an. Eine zeitgemäße Samplingfrequenz wären 44.1 kHz, da somit die Kompatibilität zur Audio CD gegeben wäre. Ein Herunterkonvertieren auf eine geringere Sampleauflösung aus Speicherplatzersparnis ist natürlich auch möglich.

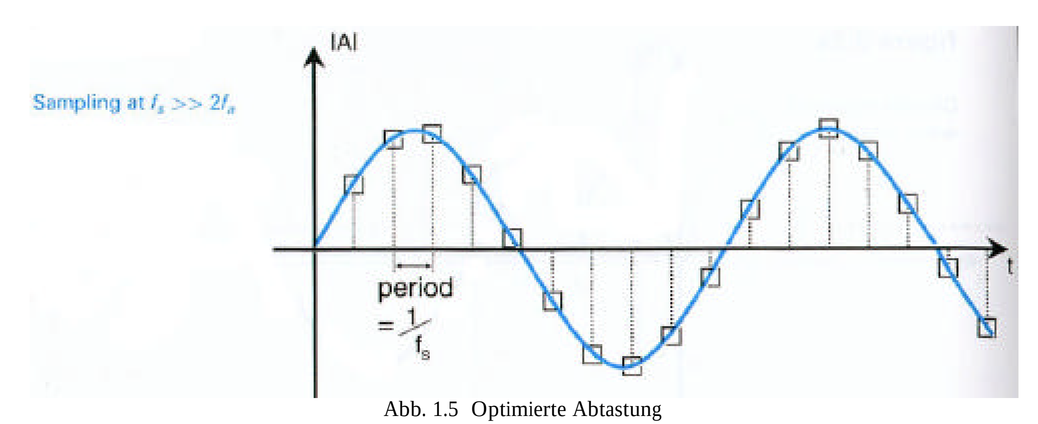


Abb. 1.5 Optimierte Abtastung

Kapitel

2

Einrichtung und Test

„Wir können erst dann bauen, wenn der Grundstein gelegt ist. Immer wenn wir ein Problem meistern, bauen wir an unserem Fundament weiter.“

Charles B. Newcomb

Hardware Anforderungen

- IBM kompatibler PC mit <Platzhalter> freiem Festplattenplatz, ein CD-Rom Laufwerk und mindestens 640kB Hauptspeicher. Ein asynchroner RS-232 serieller Kommunikations- Port (COM- Port) wird ebenfalls benötigt

- Eine 9V @ 250mA (oder höher) Stromversorgung (im Lieferumfang)

- Ein RS-232 (DB9) Kabel zum Anschluss der DSP- Karte an einen seriellen Port des PCs

- Verbindungskabel vom analogen Ein- / Ausgang zur Signalquelle bzw. zur Signalausgabe. Empfohlen wird ein Cinch- Kabel, wobei das eine Ende an einer Signalquelle, z.B. einem CD Player und das andere, zur Ausgabe auf den Lautsprechern an einen Verstärker angeschlossen wird.

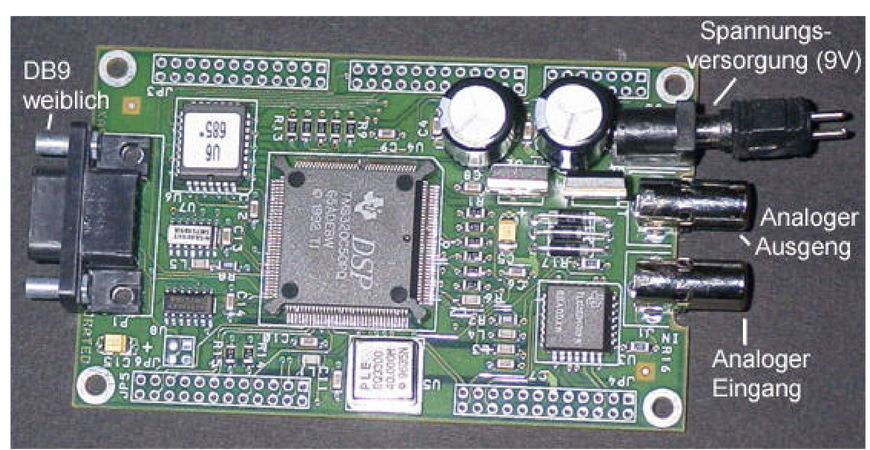


Abb.2.1 Die DSP Karte

Software Anforderungen

- MS-DOS 4.01 oder höher
- Optional Win9x / ME / NT / 2000

Anschluss der DSP Karte an den PC

1. Schalten Sie Ihren Computer aus.
2. Verbinden Sie das serielle Kabel (RS-232) mit einen COM- Port Ihres Rechners.
3. Verbinden Sie das serielle Kabel (falls nötig mit 25-zu-9 Pin Adapter) an den DB9 Sockel Ihrer DSP- Karte. Beachten Sie untenstehende Abbildung
4. Verbinden Sie den 9V Trafo mit Ihrer DSP- Karte und stecken Sie ihn in eine Steckdose.
5. Schalten Sie nun Ihren PC an.

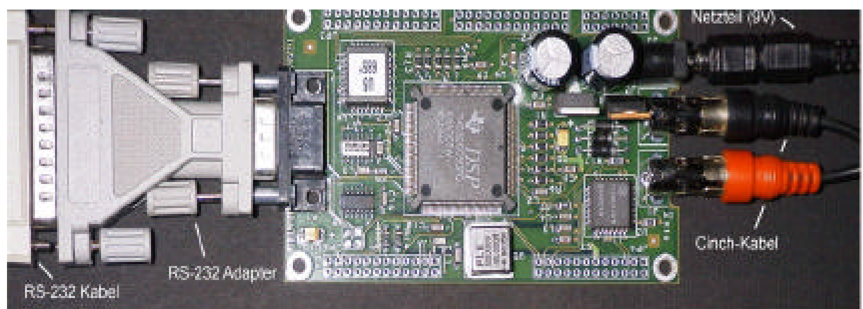
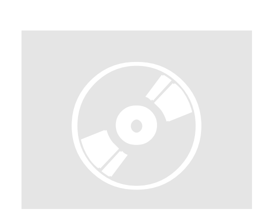


Abb. 2.2 Die DSP Karte mit allen Anschlüssen

Installieren der DSK Software



Allgemeine Installationshinweise

1. Kopieren Sie den Ordner **\DSKTOOLS** von der beiliegenden CD-Rom auf Ihre Festplatte.
2. Wenn Sie den Debugger benutzen, können Sie maximal 20 Dateien gleichzeitig geöffnet haben daher könnte es nötig sein, dass Sie Ihre config.sys modifizieren müssen.

   Fügen Sie bitte Ihrer config.sys Datei bitte folgenden Eintrag hinzu:

   FILES=20

   Nach editieren der Datei müssen Sie Ihren Computer neu starten.

   Wir empfehlen diesen Schritt nur durchzuführen falls Probleme auftreten.

3. Um sicher zu gehen, dass der Debugger korrekt arbeitet müssen Sie das PATH- Statement in der autoexec.bat (bzw. auf NT basierten Systemen, oder Win ME die Umgebungsvariablen) modifizieren.

DOS, Win 3.x, Win 9x

Fügen Sie eine neue Umgebungsvariable hinzu indem Sie folgende Zeile der Datei autoexec.bat hinzufügen:

PATH=C:\dsktools (bzw. das Verzeichnis das Sie angelegt haben)

Sie können auch einen Eintrag zu einer vorhandenen Umgebungsvariable hinzufügen indem Sie existierende Einträge mit einem Semikolon trennen. Solch ein Eintrag könnte folgendermaßen aussehen:

PATH=C:\windows\system;**C:\dsktools**

Es ist nicht unbedingt nötig nach diesem Vorgang den Computer neu zu starten. In diesem Fall müssen Sie jedoch die Datei autoexec.bat ausführen, damit das Betriebsystem den neu angelegten Pfad erkennt.

Win NT, Win 2000

Öffnen Sie die Systemsteuerung. Doppelklicken Sie auf das Symbol *System* und wählen Sie hier das Register „Umgebungsvariablen“. Hier fügen Sie wie in Unterkapitel **DOS, Win 3.x, Win 9x** beschrieben das Verzeichnis \dsktools hinzu.

Win ME

Klicken Sie auf *Start* und dann auf *Ausführen...*. In das erscheinende Feld geben Sie **msconfig** ein und drücken OK. In dem folgenden Fenster wählen Sie das Register *Umgebung* und fügen wie in Unterkapitel **DOS, Win 3.x, Win 9x** beschrieben das Verzeichnis \dsktools hinzu.

Testen der Installation

Um zu überprüfen, ob die Installation erfolgreich durchgeführt wurde tippen Sie in der Eingabeaufforderung folgenden Befehl ein und bestätigen ihn mit ENTER:

dsk5d c1

Anm.: c1 steht hier für den gewählten COM- Port. Sollten Sie einen anderen COM-Port benutzen ersetzen Sie die 1 mit der Ziffer des gewählten COM- Ports. Zum Beispiel:

dsk5d c4 für COM- Port 4.

Mögliche Fehlerquellen



- Die Baud Einstellungen könnten inkorrekt sein. Überprüfen Sie die Einstellungen des verwendeten COM- Ports in der Systemsteuerung. Die Einstellung der Baud- Rate muss der des Debuggers entsprechen.

  Mögliche Baud- Raten sind im Folgenden aufgelistet:

  - 4800
  - 9600
  - 19200
  - 38400
  - 57600

- Falls Probleme mit hohen Baud- Raten auftreten, versuchen Sie niedrigere Werte. Ein solches Problem lässt sich an einer unterbrochenen oder verrauschten Ausgabe erkennen.

- Kabel könnten nicht korrekt an die DSP- Karte angeschlossen sein. Überprüfen Sie die Kontakte.

- Die 9V Stromversorgung könnte unterbrochen sein. Überprüfen Sie ob das Netzteil korrekt an eine 230V / 50Hz Steckdose angeschlossen ist.



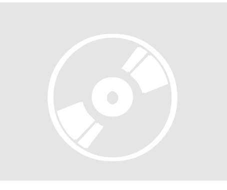
Erstellung und Ausführung mit dem DSK

„Das große Ziel des Lebens ist nicht Wissen, sondern Handlung."

Thomas Henry Huxley

Das DSK bietet dem Benutzer drei Werkzeuge mit denen er die Erstellung und Ausführung von Assemblercode bewältigen kann: Assembler, Loader und Debugger. Diese drei Programme stellen in der Assemblerwelt in vereinfachte Form ein Pendant zum Java Development Kit dar. Benutzen Sie das JDK ohne Entwicklungsumgebung, sind Sie genauso auf Editor und Eingabeaufforderung angewiesen wie beim Verwenden dieser drei Tools.

Assembler



Der Assembler dient der Übersetzung von Assemblercode in Maschinencode Object Dateien für die TMS320C5x DSP Familie . Das kommandozeilenorientierte Tool finden Sie im Verzeichnis \DSKTOOLS und wird mit dem Befehl: dsk5a <dateiname.asm> ausgeführt.

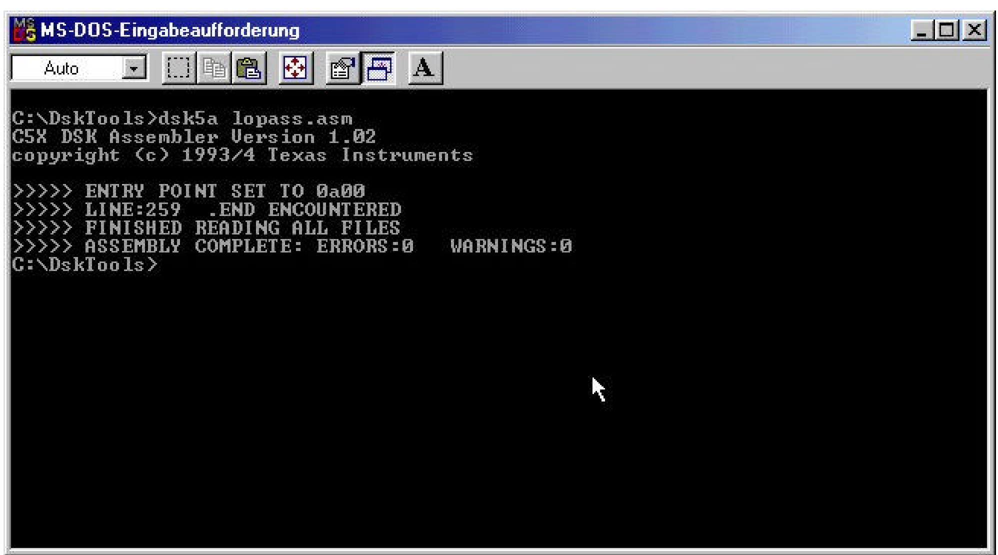


Abb. 3.1 Bildschirmausgabe nach Aufruf des Assemblers

Das Ergebnis ist eine Datei vom Typ *.dsk, die den Maschinencode enthält und vom Debugger und Loader geladen und weiterverarbeitet werden kann.

Debugger

Mit dem Debugger werden die vom Assembler erstellten *.dsk Dateien visualisiert und können von hier aus auf die DSP Karte transferiert werden.

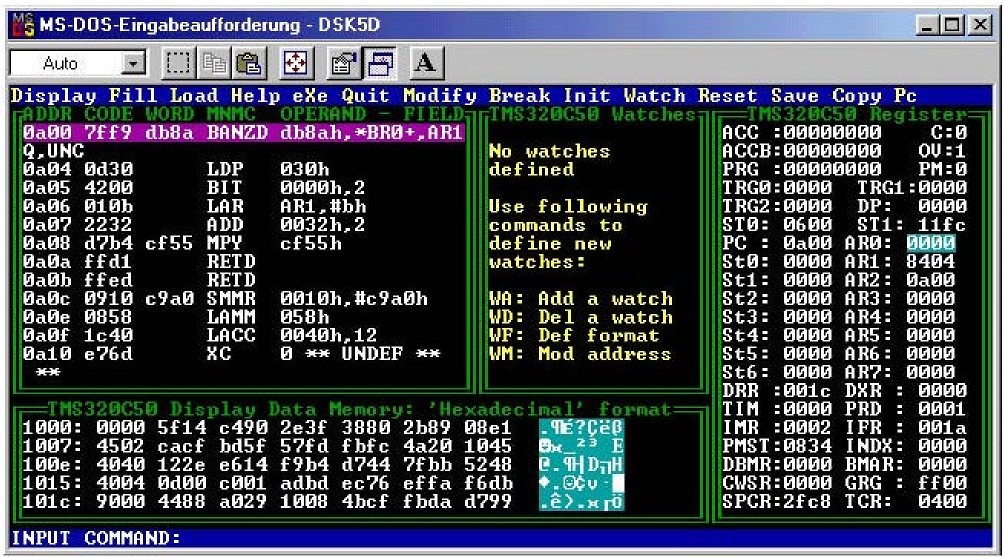


Abb. 3.2 Debuggerfenster

Loader

Der Loader ist ebenfalls ein Programm zur Ausführung der erstellten *.dsk Dateien. Im Gegensatz zum Debugger kann der Loader den Programmablauf jedoch nicht visualisieren, sondern lädt das Programm und führt dieses dann sofort aus.

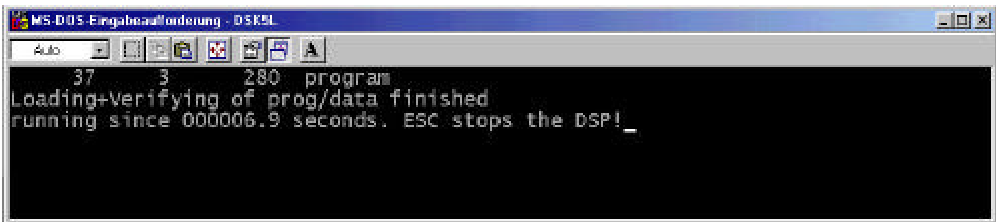


Abb. 3.3 Ausführung eines dsk-Files nach Aufruf des Loaders.

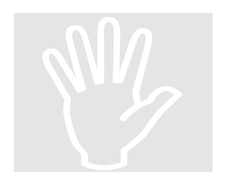
Eine genaure Beschreibung der genannten Programme folgt im Kapitel 4 bzw. im Kapitel 5, da an dieser Stelle ein praktisches Vorgehen besser verdeutlicht werden kann.

Kapitel 4

Ein Beispiel

„Niemand weiß, was er kann, bis er es probiert hat.“

Publilius Syrus



In diesem Kapitel wird anhand eines Beispiels die Funktionalität des DSK verständlich nahe gebracht. Doch Vorsicht, dieses Kapitel gibt nur Aufschluss über die Funktionalität eines einzigen Programms – des Low- Pass Filters. Es soll Einsteiger in die Materie des *Digital Signal Processing* einführen, ist jedoch kein Benutzerhandbuch für jegliche Art von DSP Projekt

Filter

Digitale Filter waren lange Zeit die am meisten gebräuchliche Anwendung für DSPs. Sie dienen dazu bestimmte Frequenzbereiche eines Signals auszuschließen. Gegenüber analogen Filtern haben digitale Filter den Vorteil, dass sie leicht reproduzierbar und modifizierbar sind, da sie im Gegensatz zu den analogen Filtern nicht fest verdrahtet sind. Ein analoger Filter ist nur begrenzt einsetzbar, da die starre Verdrahtung keinen Spielraum für Änderungen an der Filtercharakteristik zulässt. DSPs hingegen sind programmierbar und daher sehr flexibel einsetzbar. Um die Charakteristik eines digitalen Filters zu verändern bedarf es nur der Anpassung einiger Koeffizienten.

Der hier verwendete Low-Pass Filter schneidet alle Frequenzen über einer bestimmten Trennfrequenz ab, während alle darunter liegenden Frequenzen den Filter ungehindert passieren.

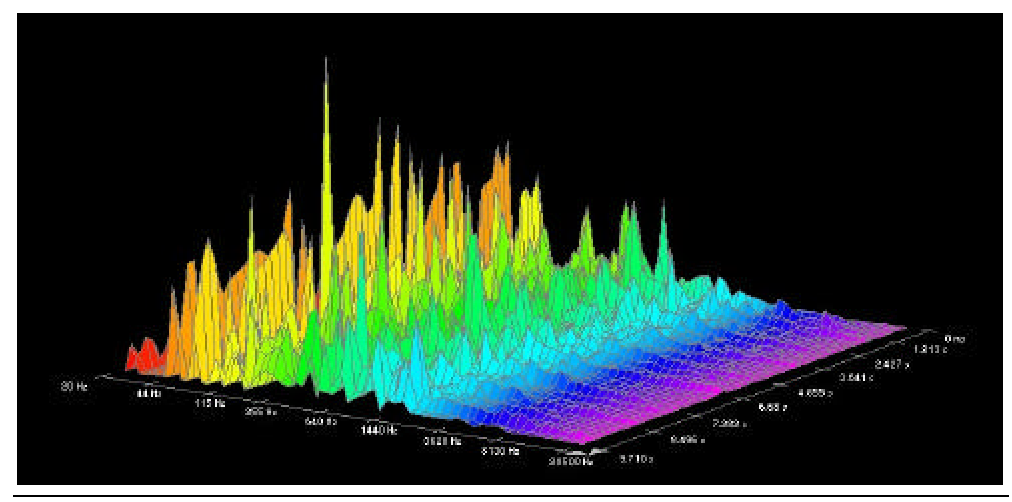


Abb. 4.1 Frequenzanalyse des ungefilterten Musikstücks

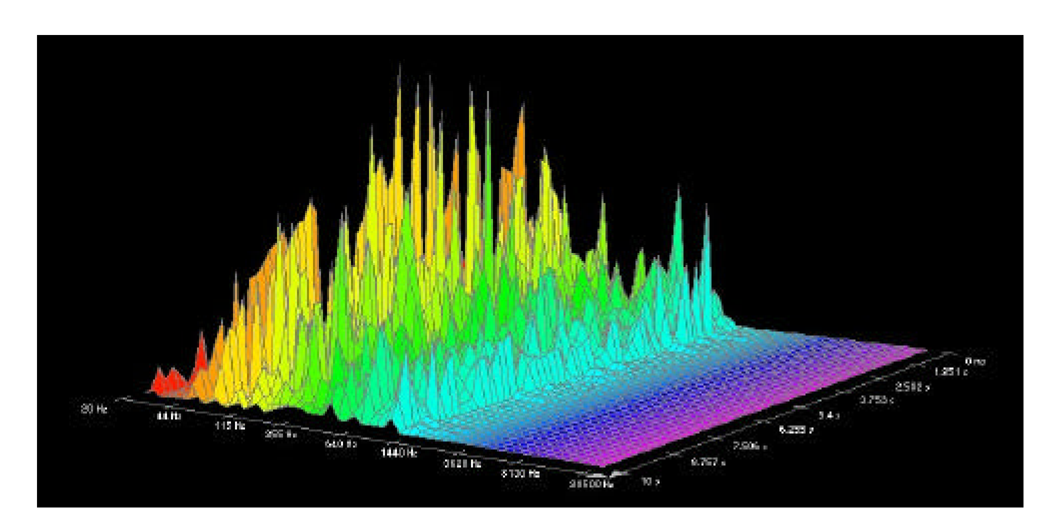
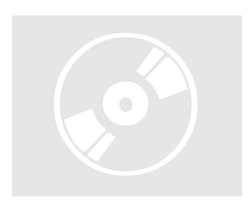


Abb. 4.2 Frequenzanalyse des gefilterten Musikstücks

Wie hier sehr anschaulich zu sehen ist, werden alle Frequenzen oberhalb der 1kHz Grenze abgeschnitten, wobei die niedrigeren Frequenzen den Filter passieren dürfen.

Das LoPass Filter Beispiel

Wie oben bereits erwähnt schneidet der Low-Pass Filter alle Frequenzen oberhalb einer bestimmten Trennfrequenz ab. Wie aber wird bewerkstelligt, dass der DSP genau dies macht?



Im Verzeichnis \DSKTOOLS\BEISPIELE auf der beiliegenden CD sind einige Beispiel- Programme zu finden, im Unterverzeichnis \FILTER auch die Datei lopass.asm. Diese Datei enthält den Assembler Quellcode für einen Low- Pass Filter und kann ohne weiteres mit einem normalen Editor wie Notepad, oder dem auf der CD befindlichen Editor (im Verzeichnis \EDITOR) angeschaut und editiert werden. Bevor Sie jedoch die Experimentierfreudigkeit packt und an der Originalstruktur des Codes Änderungen vornehmen, müssen Sie zuerst einige Vorbereitungen treffen.

Um das Programm überhaupt testen zu können, verbinden Sie den Eingang ihrer DSP Karte mit einer Klangquelle. Geeignete Signalerzeuger sind z.B. CD-Player, Keyboard oder einfach der Ausgang Ihrer Soundkarte

Den analogen Ausgang der Karte verbinden Sie entweder mit einem Aktiv-Lautsprecher, dem Verstärker Ihrer HiFi- Anlage oder dem Line-In Ihrer Soundkarte.

Damit das Programm die Signale welche am Analog- Eingang anliegen für den Filtervorgang verwendet, muss zuerst der Assemblercode angepasst werden.

In Zeile 181 finden Sie die durch ein Semikolon ausdokumentierte Programmanweisung:



;LAMM DRR ; load accumulator with word received from AIC

Es ist unbedingt notwendig das erstangeführte Semikolon zu löschen, da sonst das Eingabesignal, welches an dem analogen Cinch- Eingang anliegt, von der DSP-Karte nicht berücksichtigt wird. Nachdem Sie diese Änderung vorgenommen haben, speichern Sie die Datei und schließen vorerst den Editor.

Zum Assemblieren öffnen Sie die Eingabeaufforderung und wechseln in das Verzeichnis in dem die Beispielprogramme installiert sind. Rufen Sie auf der Kommandozeile den Assembler mit folgendem Befehl auf:

dsk5a lopass.asm

Die Ausgabe des Programms sollte mit der in *Abb. 3.1* identisch sein.



Der Assembler ist ein Programm ähnlich des Java Compilers. Der Unterschied besteht jedoch darin, dass der Java Compiler das Java Quellfile in Bytecode kompiliert und die Virtual Machine diesen in den Maschinencode umwandelt den das jeweilige Betriebssystem versteht. Da wir nun sowieso schon in Assembler Programmieren fällt das Kompilieren weg. Der Texas Instruments Assembler muss den Assemblercode nur noch in Maschinencode umwandeln.

Nun ist die Datei kompiliert, d.h. in Maschinencode übersetzt und in der Datei lopass.dsk gespeichert. Um das Programm auszuführen starten Sie entweder den Debugger oder benutzen den Loader. Der Einfachheit halber verwenden wir zuerst den Loader – auf den Debugger werden wir an späterer Stelle eingehen.

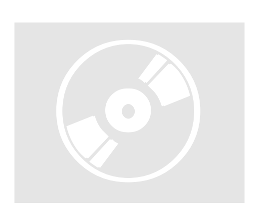


Der Loader stellt ein praktisches Tool dar, um den assemblierten Code in den Programm Speicher der DSP- Karte zu laden und sofort auszuführen. Es besteht beim Einsatz des Loaders jedoch nicht die Möglichkeit während der Laufzeit auf den Code Einfluss zu nehmen ohne die verwendete Datei neu assemblieren zu müssen.

Um den Loader mit dem erstellten lopass.dsk File zu starten muss unbedingt zuerst der verwendete COM- Port (z.B. -c1 oder –c2) und dann die auszuführende Datei genannt werden. Mit folgendem Befehl laden Sie das kompilierte lopass.dsk mit dem Loader auf die DSP Karte, die in diesem Fall mit dem COM- Port 1 verbunden ist:

dsk5l –c1 lopass.dsk

Die Ausgabe auf dem Bildschirm sollte ähnlich der in *Abb. 3.3* sein.



Sie hören jedoch noch nichts, da noch keine Audiosignale anliegen. Auf der beiliegenden CD finden Sie im Verzeichnis \SAMPLES einige Soundsamples die Sie zum Test benutzen können. Sie können natürlich auch Ihren CD Player mit einer beliebigen CD als Soundquelle benutzen.

Starten Sie nun die Soundquelle während der Loader aktiv ist. Sie sollten nun auf Ihrem Ausgabegerät eine durch den Low- Pass Filter veränderte Version des von

Ihnen verwendeten Sounds bzw. Musikstücks hören. Falls Sie noch immer nichts hören, gehen Sie bitte die einzelnen Schritte nochmals durch und beachten Sie vor allem die mit einem Blitz ⚡ versehenen Hinweise.

Codeanalyse des Low- Pass Filters

In diesem Kapitel werden wir Sie mit dem Quellcode des Low- Pass Filters in Assembler Programmierung einführen. Der hier implementierte Filter ist als FIR (Finite Impulse Response) Filter ausgelegt, dessen Filtereigenschaften von 80 Koeffizienten beschrieben werden. Die Trennfrequenz des Filters liegt bei 1kHz. Aber lassen Sie uns zunächst die grobe Struktur des Filters betrachten:

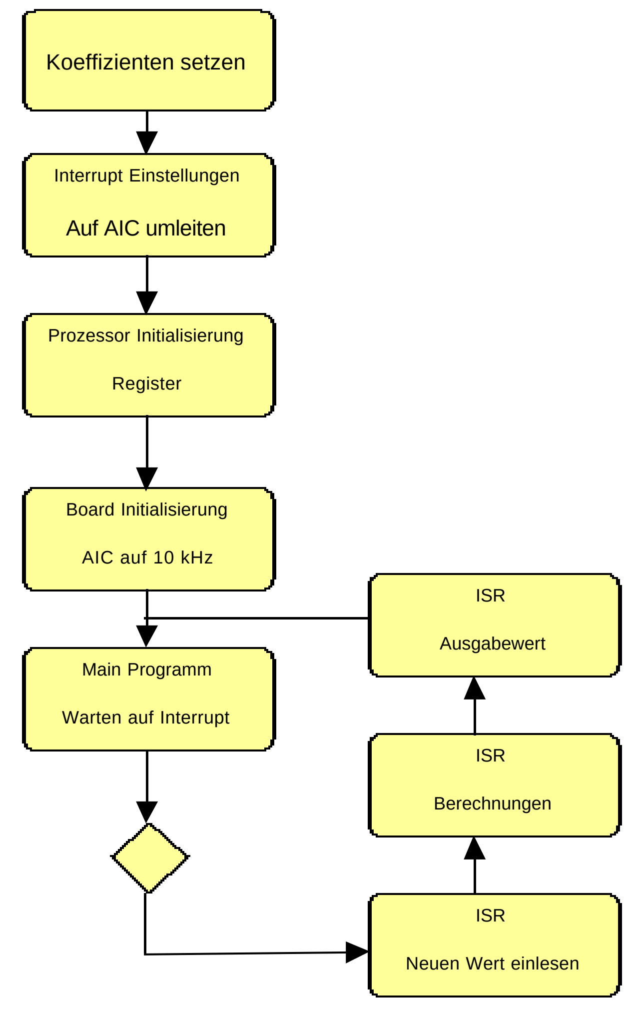


Abb. 4.3 FIR Filter Flow Chart

Öffnen Sie mit einem Editor die Datei *lopass.asm*. Um das Low- Pass Filter Beispiel lauffähig zu machen, mussten Sie bereits den Quellcode minimal verändern. Nun gehen wir aber genauer auf den Code und seine Funktionalität ein.

Zu Beginn jedes gut dokumentierten Codes findet man eine allgemeine Beschreibung über die eigentliche Funktion des Programms. So wird das auch bei Assembler Programmen gehandhabt. Erste wichtige Informationen stehen am Kopf jedes Codes. Wie bereits an anderer Stelle erwähnt erfolgt das Auskommentieren von Zeilen mittels eines Semikolons – das Pendant dazu in Java bzw. C++ wäre der doppelte Slash //.

Eine Auflistung der verschiedenen Datentypen finden Sie im Starter Kit ab S. 5-2.

Initialisierung

Der eigentliche Quellecode erfolgt ab Zeile 26 mit 2 Assemblerdirektiven. Die erste .MMREGS definiert auf globaler Ebene symbolische Namen für die TMS320 Register und platziert diese in eine Tabelle. Eine Übersicht dieser Memory-Mapped Registers finden Sie im DSP Starter Kit auf Seite 5-25.

Die nachfolgende .ds Direktive befiehlt dem Assembler den Quellcode in den Datenspeicher (Data Memory) zu laden. Die dazugehörende 16bit Speicheradresse beschreibt den Initialwert für den Programmzähler.

In den Zeilen 28-33 werden in vier verschiedene Register Zahlenwerte eingesetzt, über die sich die Samplingfrequenzen festlegen lassen. Hierbei bilden jeweils die Register TA und TB bzw. RA und RB Wertepaare für das Senden und Empfangen des Ein- bzw. Ausgangs. Die entsprechende Formel lautet:



$$Samplingfrequenz = \frac{MCLK}{TA \cdot 2 \cdot TB}$$

Das im Zähler stehende Register MCLK ist standardmäßig mit dem Wert 10 MHz belegt. Durch die im Quellcode zugewiesenen Werte TA=16 und TB=31 ergibt sich eine Samplingfrequenz von 10 kHz.

In Zeile 34 wird dem AIC_CTR (Analog Interface Circuit_Contol Register Interrupt) eine Adresse zugewiesen. Der AIC ist der Bezeichner der analogen Wandlersektion.

Darauffolgend wird ein temporärer Speicherbereich mit dem Namen OUTPUT reserviert und initialisiert. Dieser Bereich wird für das Speichern der Ausgangssignale des Filters verwendet.

Die Speicherbereiche TEMP, TEMP1 und die Konstante seed werden von dem Random Noise Generator benutzt, der am Anfang des Quellcodes kurz erläutert wird, hier aber nicht weiter beschrieben werden soll.

Im nächsten Block des Quellcodes finden sie die vom Filter Coefficient Generator erzeugten Filter Koeffizienten. Diese 80 Koeffizienten werden in den Datenspeicher gelesen und mit jeweils einem Bezeichner, h0 bis h79, für die Referenzzierung versehen. Die Koeffizienten werden im Q15 Format gespeichert, wobei rechts neben dem Semikolon ihr Dezimalwert angegeben wird. Beachten Sie, dass die Koeffizienten ein symmetrisches Verhalten zur Mitte hin aufweisen (vergleichen Sie in diesem Zusammenhang z.B. die Zeilen 80 und 82).

In den Zeilen 124 bis 132 – XN bis XNLAST – werden 80 Speicherbereiche für die Eingangssamples reserviert und initialisiert.

Hauptprogramm

Das Filtern findet in der sogenannten ISR (Interrupt Service Routine) statt. Diese Routine wird jedes Mal aufgerufen wenn ein neues Sample vom AIC digitalisiert wird. Nachdem wir die Samplingfrequenz auf 10 kHz gesetzt haben, könne wir davon ausgehen, dass diese Routine 10.000 mal pro Sekunde bereitgestellt wird. Die Gleichung zur Berechnung der Filterausgabe verwendet die oben beschriebenen 80 Koeffizienten:

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) + .. + h_{78} x(n-78) + h_{79} x(n-79)$$

Die ISR implementiert diese Gleichung.

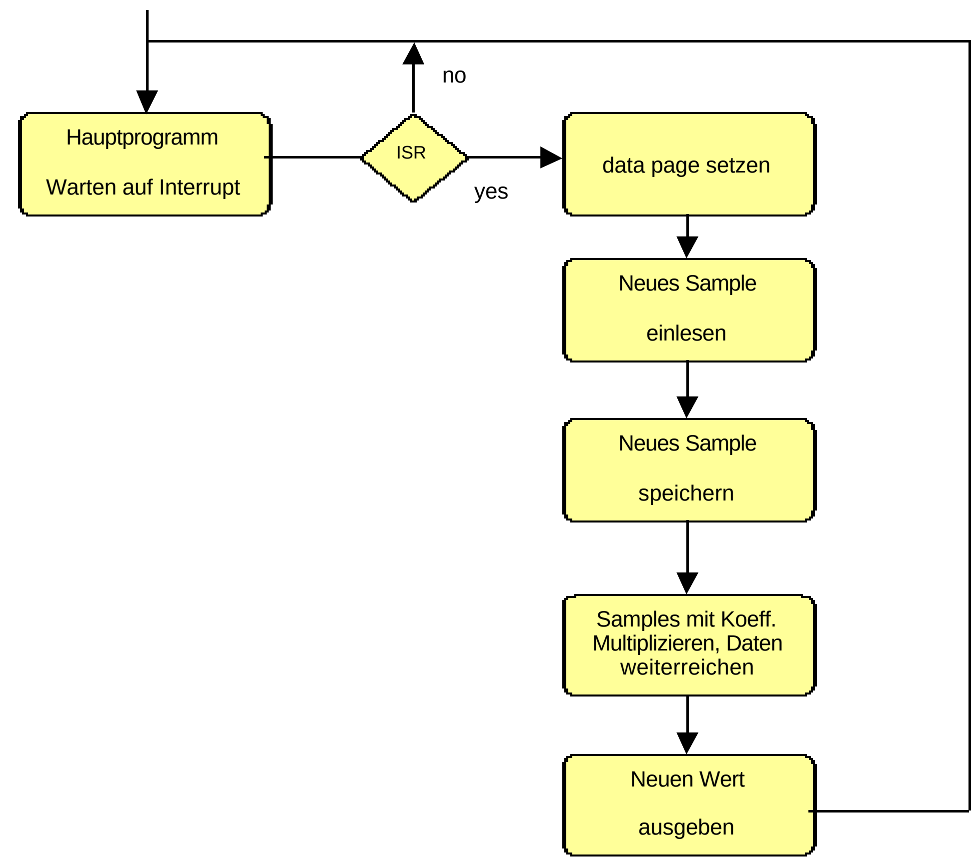


Abb. 4.4 FIR ISR Flow Chart

Im Quellcode finden Sie die Interrupt Service Routine ab Zeile 142 bis Zeile 197. Innerhalb dieses Blocks finden Sie auch die Implementierung des Random Noise Generators, der jedoch auskommentiert ist und nicht weiter beachtet werden muss. Die SETC Instruktion in Zeile 142 setzt das Control Bit auf den Interrupt, um eine Ablaufsteuerung auf Basis des AIC Interrupts zu erreichen. Dieser Modus wird in Zeile 155 durch die Anweisung CLRC beendet und der Interrupt dadurch wieder freigegeben.

In Zeile 143 wird der Data Memory Page Pointer mit dem Wert 0 adressiert (LDP).

Die darauffolgenden Zeilen dienen einerseits dazu weitere Initialisierungen auszuführen was insbesondere den AIC betrifft: Darüber hinaus befindet sich an dieser Stelle auch die Implementierung des Random Noise Generators.

Die nächste den ISR betreffende Codezeile befindet sich in Zeile 181. Wie bereits oben angedeutet wird hier aus dem Data Receive Register (DDR) das neuste Sample in den Akkumulator geladen. Das ist unser Sample: x(n). In der nächsten Zeile wird dieser Wert in der dazugehörigen Variablen XN gespeichert.

In Zeile 184 wird das älteste Sample in ein Auxiliary Register herausgeschrieben (LAR), danach werden der Akkumulator und das Produktregister gelöscht (ZAP), was die Vorbereitung für eine Multiply/Accumulate Schleife beschleunigt. Nachfolgend wird das Auxiliary Register AR0 als aktuelle Auxiliary Register gesetzt (MAR).

In den nächsten drei Zeilen wird der Befehl zum 80fachen Wiederholen (RPT) der Anweisung MACD gegeben, welche die folgenden fünf Schritte pro Durchlauf enthält...

1. Schreibe den Inhalt des Produktregisters in den Akkumulator (zu Beginn 0).

2. Multipliziere den Wert, der in h0 steht mit hx (zu Beginn AR0 = XNLAST, hx = h0).

3. Kopiere den Bereich auf den AR0 zeigt in den darauffolgenden Bereich (zu Beginn XNLAST → XNLAST+1; zum Schluss XN → XN+1).

4. Dekrementiere die Adresse in AR0 um 1 (*-).

5. Inkrementiere die Adresse in h0 um 1 (wird durch Schleifenwiederholung erzeugt).

Alle Samples werden jedes Mal wenn die Instruktion ausgeführt wird mit den dazugehörigen Konstanten multipliziert und zum Akkumulator dazuaddiert. Die Instruktion verschiebt außerdem alle Samples um einen Speicherbereich. Dadurch ist der Filter wieder bereit wenn das nächste Sample ankommt, was auch der Grund dafür ist, dass AR0 auf das letzte Sample zeigt, anstatt auf das erste. Eine solche Manipulation wäre nicht möglich, wenn die Filterkoeffizienten nicht symmetrisch wären.

Das Ergebnis dieser sukzessiven Multiplikation und Addition werden im Akkumulator im Q30 Format gehalten. Der höhere Akkumulator repräsentiert den Integerteil der Nummer, welche mit einem left-shift in den temporären Speicherbereich Output gespeichert wird. Der left-shift eliminiert das Vorzeichen-Bit. Die niedrigeren 16 Bit werden nicht berücksichtigt, was einen Effekt auf die Genauigkeit des Filterns hat, jedoch nicht signifikant ist. Zu finden ist diese Instruktion in der Zeile 193 SACH OUTPUT,1.

In der nächsten Zeile wird der Inhalt von Output wieder in den niedrigeren Akkumulator geladen (LACC). Die Instruktion SFL führt einen left-shift auf den Akkumulator aus. Diese Verschiebung um ein Bit vereinheitlicht den Inhalt ins Q15 Format., was gleichbedeutend mit einer Division von 2^{15} ist.

In der nächsten Zeile werden die beiden Least Significant Bits (LSB's) auf Null gesetzt, da der AIC nur 14 Bit Signale verarbeiten kann (AND). Anschließend wird das Datenwort ins Data Transmit Register (DXR) geschrieben (SAMM). Mit RETE wird der Block wieder verlassen.

Diese Beschreibung sollte ausreichen um die grundlegende Funktionalität des FIR Low- Pass Filters zu erfassen. Im selben Verzeichnis wie der Low- Pass Filter finden Sie die Quellcodes für High- Pass, Bandpass und Bandstop Filter

Kapitel

5

Der Debugger

„Es ist schwerer, einen guten Stuhl zu bauen als einen Wolkenkratzer.“

Ludwig Mies van der Rohe

Der DSK Debugger ist ein einfach zu lernendes und zu benutzendes Tool dessen fensterorientierte Benutzeroberfläche (*Abb. 5.1*) die Einarbeitungszeit verkürzt und das Auswendiglernen von komplexen Kommandos überflüssig macht. Der Debugger kann Quellcode laden und ausführen, mit der zusätzlichen Möglichkeit der Einzelschrittausführung (single-step) und dem Setzen von break-points an denen das ausgeführte Programm anhält.

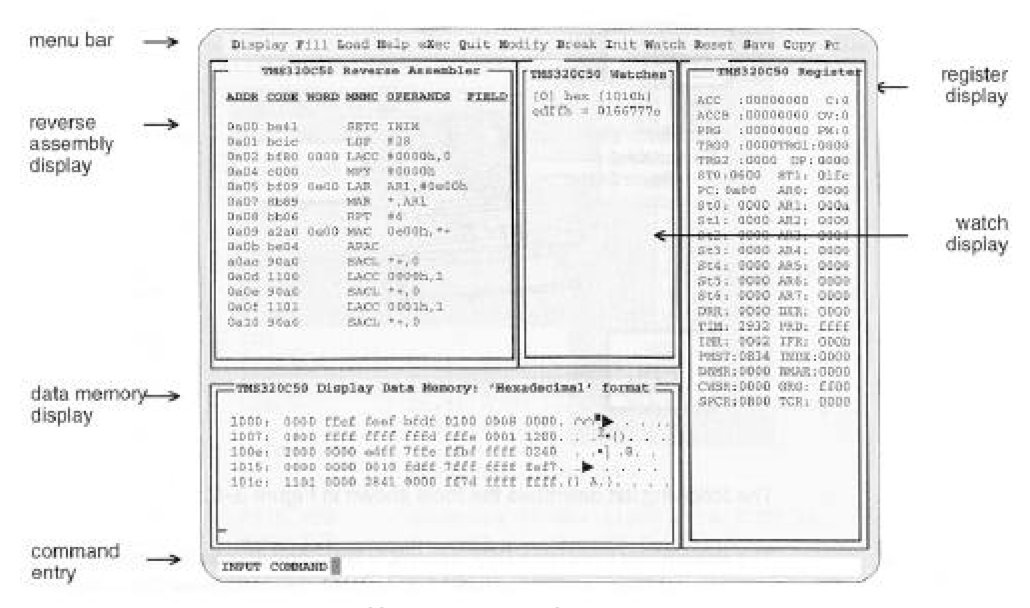
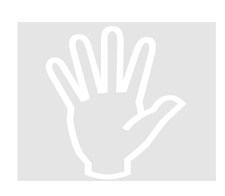


Abb. 5.1 Das DSK Debugger Fenster

Zwei Beispiele



Data Memory mit hex pattern füllen

Die Menüleiste zeigt die aktuell zur Auswahl stehenden Kommandos. Alle Menüpunkte haben Untermenüs, wobei die meisten Kommandos nicht mehr als eine Folge von zwei Buchstaben zum Ausführen brauchen. Als Shortcuts zu den einzelnen Kommandos dienen die fett geschriebenen Großbuchstaben in der Menüleiste. Eingegebene Kommandos werden in der Kommandoleiste am unteren Bildschirmrand angezeigt.

Um zum Beispiel einen Speicherbereich anzusehen muss das *Display* Menü mit „D" aktiviert werden, womit dessen Untermenü angezeigt wird. Wählen Sie in diesem Untermenü abermals „D" – welches diesmal für *Data* steht – und es wird der Inhalt des Data Memorys in der unteren Hälfte des Bildschirms angezeigt. Geben Sie „FD" für *Fill \ Data* in die INPUT COMMAND Zeile ein und bestätigen Sie mit ENTER, worauf ein Dialogfenster erscheint. In diesem Dialogfenster geben Sie als *Start Address* 1000, für *Length* 7f und für *Fill pattern* abcd ein. Sobald die Nachricht *Fill Data Memory finished* erscheint drücken Sie Esc um so das Dialogfeld zu verlassen. Im unteren Fenster werden nun die gefüllten Speicherbereiche dargestellt. 127 (7f) Speicherstellen der Länge eines Words (16 bit) sind nun mit dem hex pattern abcd gefüllt worden.

Verändern eines Registerinhalts

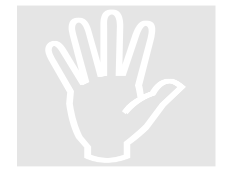
Geben Sie „MR" für *Modify \ Register* in die INPUT COMMAND Zeile ein und bestätigen Sie mit ENTER, worauf ein Dialogfenster erscheint. Auf der Eingabezeile des Dialogs geben Sie folgende Anweisung ein:

accu = 1f1f

Dies wird den Inhalt des Akkumulators verändern. Der Akkumulator ist für 32 bit Werte ausgelegt, wir haben jedoch nur einen 16 bit Wert eingegeben. Überprüfen Sie nun die Einträge im rechten Fenster, in dem die Register angezeigt werden. Sie werden sehen, dass sich der Inhalt von ACC in 00001f1f geändert hat (32 bit). Alle Register in diesem Fenster haben ihre Standardnamen. St0 bis St6 zeigen den Hardware Stack womit Sie die Operationen des Stacks überwachen können, wenn eine CALL Anweisung ausgeführt wird.

Es ist empfehlenswert mit einigen Kommandos herumzuspielen bevor Sie in diesem Dokument fortfahren. Benutzen Sie zum Beispiel „MD" um den Inhalt einer Speicherstelle zu verändern, wobei Sie den veränderten Inhalt im Data Display Fenster sehen können. Außerdem können Sie mit „WA" (*Watch \ Add*) Wächter für diese Speicherstellen definieren und Änderungen im watch display überwachen (siehe *Abb.* 5.1).

Debugging des Low- Pass Filters



Um Ihnen anhand des vorherigen Low- Pass Filters auch noch eine sinnvolle Verwendung des Debuggers aufzuzeigen möchten wir in diesem Abschnitt aus dem Debugger heraus die Samplingfrequenz der Ein- und Ausgänge der DSP- Karte verändern.

Da die Register des AIC nicht direkt aus dem Debugger angesprochen werden können, müssen Sie anstelle des Registernamens dessen entsprechenden Speicherstellen im Data Memory verwenden. Um nicht mühsam im Code nach den entsprechenden Stellen suchen zu müssen, erstellen Sie mit dem Assembler ein List File (*.lst), welches die Quellanweisungen mit der entsprechenden Zeilennummer, der Speicheradresse, dem in der Speicheradresse befindlichen Wert, sowie unaufgelöste Symbole enthält.
Das List File in unserem Beispiel erstellen Sie mit folgendem Befehl in der Eingabeaufforderung:

dsk5a lopass.asm –l

Öffnen Sie nun die eben erstellte Datei lopass.lst mit einem Editor. Wie bereits im vorherigen Kapitel erläutert definiert sich die Samplingfrequenz über die AIC Register TA, TB, RA und RB. Suchen Sie in der List Datei nach den entsprechenden Einträgen für die Register im oberen Abschnitt des Codes.

Ein solcher Eintrag eines dieser Register könnte folgendermaßen aussehen:

```
00028 0f00 0010  TA       .word    16
```

Die erste Spalte gibt die Zeilennummer an (00028), während in der zweiten die Speicheradresse für diese Anweisung steht (0f00). Die dritte Spalte gibt den Inhalt dieser Speicheradresse als Hex Wert an (0010). Die drei darauf folgenden Spalten sind original vom Quellfile übernommen worden (siehe lopass.asm Zeile 28).

Nun können Sie sich die Speicheradressen für die oben genannten Register notieren und den Editor verlassen. Wechseln Sie jetzt wieder in das Debugger Fenster um die Register für die Samplingfrequenz zu verändern. Vergessen Sie nicht ein Audiosignal am Eingang der DSP- Karte anzulegen.

Laden Sie die Datei lopass.dsk über das Kommando „LD“ in den Debugger. Definieren Sie nun sinnvollerweise Wächter für die 4 notierten Speicheradressen (0f00 bis 0f03). Im Watch Display sollten nun die vier Speicheradressen mit ihren zugehörigen Werten zu sehen sein. Die Formel zur Errechnung der Samplingfrequenz haben wir bereits im vorherigen Kapitel beschrieben. Denken Sie daran, dass die maximale Samplingfrequenz des AIC bei 19,2 kHz liegt. Um zum Beispiel die volle Samplingfrequenz von 19,2 kHz zu nutzen könnte man die Werte TA, RA = 9 und TB, RB = 29 wählen. Doch bevor Sie die Samplingfrequenz verändern, hören Sie sich besser noch einmal den Originalfilter a n, indem Sie im Debugger mit dem Kommando „XG“ das Programm starten.

Brechen Sie nun die Ausführung mit Esc ab und ändern Sie die Register wie folgt: Mit dem Kommando „MD“ können Sie den wert einer bestimmten Speicherstelle im Data Memory verändern. Geben Sie nun als Speicheradresse 0f00 für TA ein und geben Sie als neuen Wert dafür 9 ein. Nachdem Sie mit ENTER bestätigt haben, erscheint in einer neuen Zeile als editierbarer Wert die darauffolgende Adresse. Geben Sie nun nacheinander die Werte für RA, TB und RB ein. Denken Sie daran, dass die eingegebenen Werte im Hex Format eingegeben werden müssen (dezimal 29 = hex 1d). Verlassen Sie den Dialog nach Eingabe der Werte mit Esc. Wenn Sie bei den eben gemachten Änderungen keinen Fehler gemacht haben, finden Sie die geänderten Werte im Watch Display.

Damit das Programm mit den veränderten Werten ausgeführt wird ist es zwingend notwendig „I“ für Init zu drücken. Starten Sie nun das Programm über das Kommando „XG“ erneut. Das Ausgangssignal sollte nun mit hörbar verbesserter Qualität ausgegeben werden.

Nun sollten Sie mit einigen grundlegenden Funktionen des Debuggers vertraut sein um ein eigenes Projekt qualitativ hochwertig und effizient bearbeiten zu können.



Abschließende Worte

„Nach dem Spiel ist vor dem Spiel."

Sepp Herberger

Zu guter letzt möchten wir an dieser Stelle noch einige abschließende, mitunter auch persönliche Worte an den Leser richten. Wir hoffen sehr, dass dieses Handbuch seinen Zweck erfüllt hat und einige Stolpersteine auf die wir während der Erstellung dieses Dokuments stießen, beseitigt werden konnten. Natürlich konnten wir in der uns gegeben Zeit nicht alle Eventualitäten abdecken, aber hält man sich vor Augen, dass wir allein für dieses Projekt einen Aufwand von ca. 70 Stunden pro Person erbrachten, so sollte es doch zu einiger Zeitersparnis in folgenden „Generationen" führen.

Auf der beiliegenden CD finden Sie neben den für das Arbeiten mit der DSP- Karte lebenswichtigen Programmen wie Assembler oder Debugger auch einige nützliche Tools wie den Adobe Acrobat Reader 4.0 oder den MP3 Player WinAmp 2.64. Bei diesen Tools handelt es sich um Free- oder Shareware – wir bitten mit diesen Programmen keinen Missbrauch in irgendeiner Form zu betreiben.
Im Unterverzeichnis \SAMPLES\BEISPIELSOUND finden Sie ein kleines MP3 File mit dem wir unser gesamtes Low- Pass Filter Beispiel testeten.

Dieses Projekt ist für uns nun schließlich erledigt – andere Projekte verlangen nun unsere volle Aufmerksamkeit, jedoch wollen wir es uns nicht nehmen lassen einige Danksagungen auszusprechen:

- ♥ Bahlsen – für die Verringerung des Frusts mit diversem Knabbergebäck
- ♥ CocaCola – für den ständigen Nachschub von Zucker und Koffein
- ♥ harman/kardon – für die musikalische Begleitung
- ♥ Microsoft – für die Entwicklung einer so schnellen Textverarbeitung
- 🚭 Philipp Morris – für die Beruhigung unsere Nerven
  Rauchen gefährdet Ihre Gesundheit!

In diesem Sinne wünschen wir ein erfolgreiches Arbeiten mit dem TMS320C50.

Mittwoch, 13. Dezember 2000
Christoph Metzendorf Michél Büchner